Finite Context Models

Teoria Algorítmica da Informação - 40752

Gonçalo Marques - 98648
João Reis - 115513
Renan Ferreira - 93168

Aveiro
May, 2024

Table of Contents

Abstract

This study discusses the development of a program, was_chatted, following concepts of finite context models, particularly based on Markov models, to determine whether a piece of text was written by a human or generated by an AI. By changing the size/amount of context, it was noted that it significantly affected precision achieving ~96% under optimal, possible conditions. Limitations on detecting what the maximum context size would be, hindered possibilities for further testing but, despite these hurdles, this approach presents itself as viable and shows potential to become better and even more effective.

Introduction

In the current landscape of the internet, AI is becoming increasingly more prevalent every day. Internet users, including students, are using artificial intelligence to do more of the writing for them and it's often hard to distinguish between a text which has been written by a human or an AI causing concerns for scholars and the general public as the information provided by AI isn't necessarily fully trustworthy. Given this context, many tools are being developed and considered (such as GPTZero) to find similarities between texts and attempt to correctly identify if these were written by humans or AI.

Normally, the approaches to classify texts, involve feature extraction and selection. These features are then put into a function that maps them onto unique classes and executes a classification process.

In this paper, we explore the idea of using compression techniques and algorithms to determine how a text was created. For each class (r_h for Human, or r_c for ChatGPT) associated with a piece of text, a model is built that encapsulates that given text. This set of data will then be used to calculate the test text's normalised relative compression using the model's entropy. The higher the relative compression compared to a model, the higher the likelihood of those being similar. This, in short, means that the model that can use the least amount of bits to describe a context/text will be the most similar to it, allowing the program to predict which class the reference text belongs to.

Methodology

Objectives

The primary objective of this project was to develop a program named “was_chatted” which is able to determine whether or not a text was written by ChatGPT. The program had to, at the very least, take, as an input, three files. Two of these files contain training data to use to make the r_h and r_c models and, the third file is a test file to check out the training results.

The program is intended to apply concepts of finite context models for text representation and creation of models. These models are then used to estimate the compressibility of the text under analysis in comparison to the reference texts from the training.

To guarantee the reliability of the training phase, a large number of texts was necessary entering the realm of megabytes of text. Datasets were gathered from Kaggle containing both AI-generated and Human-written texts for both training and testing/evaluating the model.

Arguments and Variables

The program takes in 6 arguments:

- A csv file, composed of multiple smaller texts written by humans and by ChatGPT. This is used for testing and evaluating the predictions by registering the number of hits and misses;
- A text file for experimentation to try and do a one-time prediction of a larger text whether it’s human-made or AI-written;
- The path to a .csv file, made off of texts written by humans to make a model of the r_h class;
- The path to a .csv file, made off of texts written by AI to make a model of the r_c class;
- An α which serves as a smoothing parameter (further explained later);
- And, lastly, an integer value, k, which determines the amount of characters of context for each event.

General Theory

Finite Context Models

Finite Context Models (FCMs) are based on the same ideas as Markov Models. It consists of the notion that the probability of observing a symbol/event at a given position in a sequence solely depends on a finite history of previous symbols. This finite history is determined by the order of the model, denoted as k. For instance, a first-order model will simply consider one symbol before predicting the following symbol. By increasing k, therefore increasing the order of the mode, the model will make predictions based on longer sequences and, as such, have more context. This technique is particularly good at uncovering dependencies and finding patterns in data.

Calculations

In information theory, an instrumental concept is Entropy, H, which quantifies the randomness associated with a variable, x. In the context of text processing and compression, entropy measures the average number of bits needed to represent each symbol in a text. This can be calculated with the formula:

$$H(x) =- \Sigma_i P(x_i) log_2(P(x_i)),$$

where $P(x_i)$ is the probability of each symbol occurring in the sequence.

With this understanding normalised relative compression, NRC, can now be understood. This is a metric used in this project to evaluate the compressibility of a text sequence compared to a model which will then allow the program to predict whether said text is AI-generated or human-written. For this, the entropy of the text under analysis, H_t, is subtracted from the maximum possible entropy of the model, H_m, and then this is divided by H_m resulting in the following formula:

$$NRC = \frac{H_m - H_t}{H_m}.$$

A higher NRC value will indicate better compression of a text sequence using the given model, implying a closer relationship between the text under analysis and the model. On the other hand, a lower value of NRC shows a poorer match between the text and the model's learned patterns.

The program

The FCM program takes in the previously mentioned variables and reads through the files, storing the number of times an event happened after each possible context of k length. Then, it calculates the probability of these events happening and makes a calculation of the entropy which showcases the number of bits, on average, needed to represent each symbol in a text.

Initially, there were issues regarding RAM for values of k bigger than 4. The program would be killed automatically by the machine as it would cap out the machine's RAM easily and more memory would be required to proceed executing the program. The original data structures used an absolute counter for the finite context model needed during the construction phase while it was being fed texts. On top of that it would use another data structure which saved the relative probability estimations which, for symbols in which the count is zero according to a certain context, would keep using the formula with the smoothing parameter. This meant that an enormous sum of data stored in memory in the end could be considered largely redundant.

The new model, a compressed version of the previous FCM, in order to reduce the amount of RAM used, does not make use of an absolute counter. This time around, the data structure which stored the probability estimations and/or relative frequencies of each model, only stores the count of each event. This means that now instead of storing multiple results from multiple calculations, it simply registers a context (if k=3 in the text “ABCD”), “ABC”, followed by the event, “D”, as +1 to the number of times “D” has occurred following “ABC” during training.

If, at any point during the calculation of a text’s entropy, the probability estimation is needed given a certain context, the program will not save the estimation for the symbol. Instead, the probability is estimated using an original function at the time of executions. To obtain said estimation, all that is necessary is to create a new data structure which is, basically, a context counter. In the end, the new data structures end up being more than 10% smaller than the original FCM design. With these changes, the program is able to run up to k=6 as it gets killed when k>6.

During the process of estimating probabilities, the smoothing parameter, α, is used to avoid attributing 0 probability to events that weren’t seen during training as this would cause them to remain 0 permanently and affect predictions.

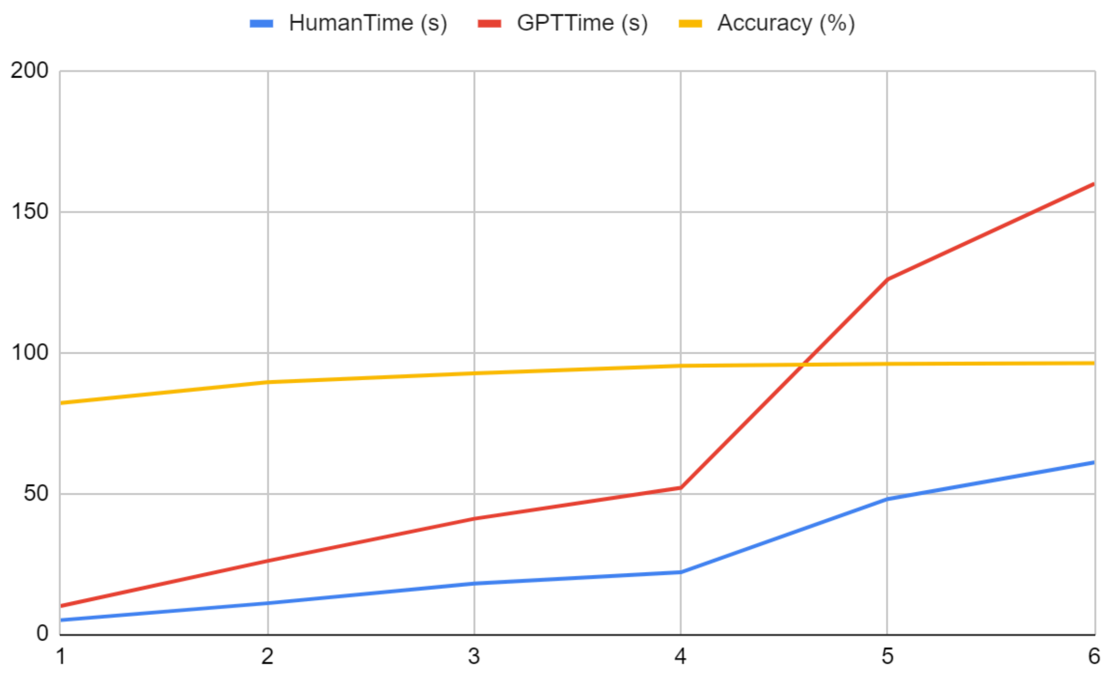
Results

Testing was done by incrementally increasing k, and using the most reliable and fastest value to test how changing α would affect the results. A notable aspect of the k values was that values of 1 or 2 can be slightly inconsistent when using more well-written texts. An excerpt from “The Great Gatsby” was used to attempt to test the program and it does incorrectly guess that it’s human-made for these values of k becoming much better and precise as k is increased.

Below, Table.1 and Graph.1 show the effects of modifying k. As it increases so does the training time however, more importantly, the accuracy increases greatly hitting a very high 96.3% with k=6. A test was also executed using the optimal parameters with the full testing dataset as a “stress” test providing a comforting 94.8% hit-rate. On the graph, the final test was removed as it affected visibility to a certain degree.

k	HumanTime (s)	GPTTime (s)	Total	Hits	Misses	Accuracy (%)	TextGuess
1	5	10	1147	942	205	82.1	Incorrect
2	11	26	1147	1026	121	89.5	Incorrect
3	18	41	1146	1062	84	92.7	Correct
4	22	52	1145	1091	54	95.3	Correct
5	48	126	1144	1098	46	96	Correct
6	61	160	1140	1098	42	96.3	Correct
6	75	191	479884	454910	24974	94.8	Correct

Table.1 - Testing results for 0<k<7 and α=1.



Graph.1 - Testing results for 0<k<7 and α=1.

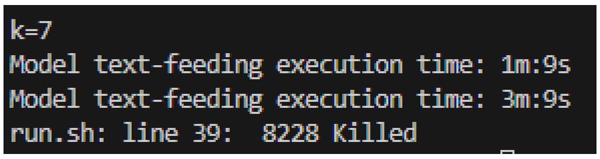
Tests with different values of α weren't as promising, decreasing performance as it was increased. For these tests, k was used with value 3 as it was the fastest value that was also reliable enough to trust the results of. Changes here did not affect runtime in any meaningful way as it is only an integer number getting its value changed in one single calculation during probability estimation. Table.2 shows the downward trend of increasing α up to 4 as it seemed useless to increase it further.

a	HumanTime (s)	GPTTime (s)	Total	Hits	Misses	Accuracy (%)	TxtGuess
1	22	51	1145	1091	54	95.3	Correct
2	23	50	1145	1087	58	94.9	Correct
3	22	50	1145	1085	60	94.8	Correct
4	22	57	1145	1084	61	94.7	Correct

Table.2 - Testing results for 0<α<5 and k=3.

Discussion

Experimentation during development provided valuable insights into the program's performance and other limitations. Increasing the number of characters in the context sequence, k, improved the accuracy. However, it wasn't possible to test where the fall-off would start as the program got killed automatically after the training phase for values of k superior to 6 (Fig.1).



```
k=7
Model text-feeding execution time: 1m:9s
Model text-feeding execution time: 3m:9s
run.sh: line 39:  8228 Killed
```

Fig.1 - Program being killed by the system after training with k=7

Through testing, when k=6 however, the program hit a maximum accuracy of 96.3% which could hardly become better. This indicates the effectiveness of the approach taken. The issues noted with small values of k also highlight the trade-off between the complexity of the model (controlled by the value of k) and its performance. It’s also important to take into consideration that the overhead of the increased computational time might not compensate for the benefits of a slightly better, more accurate model.

It was also observed that increasing k would substantially increase computation time during the training phase which initially, with the original program, would take over 10 minutes to process the ChatGPT training dataset with k=5. With the new version of the program, the compressed FCM, the time was lowered to less than 3 minutes for the whole training.

With the full dataset, it was possible to achieve just under ~95% accuracy which is still incredibly good considering the almost 500 thousand lines of text being analysed and thus, proving the reliability and consistency of the developed algorithm.

Conclusions

Despite there being some challenges regarding performance and range of tests done, the program behaved incredibly well with a maximum accuracy of ~96% which is far higher than some current commercially sold algorithms at the time of this study. We learnt plenty about finite context models and therefore the Markov model techniques which were employed in making this project.

Future Work

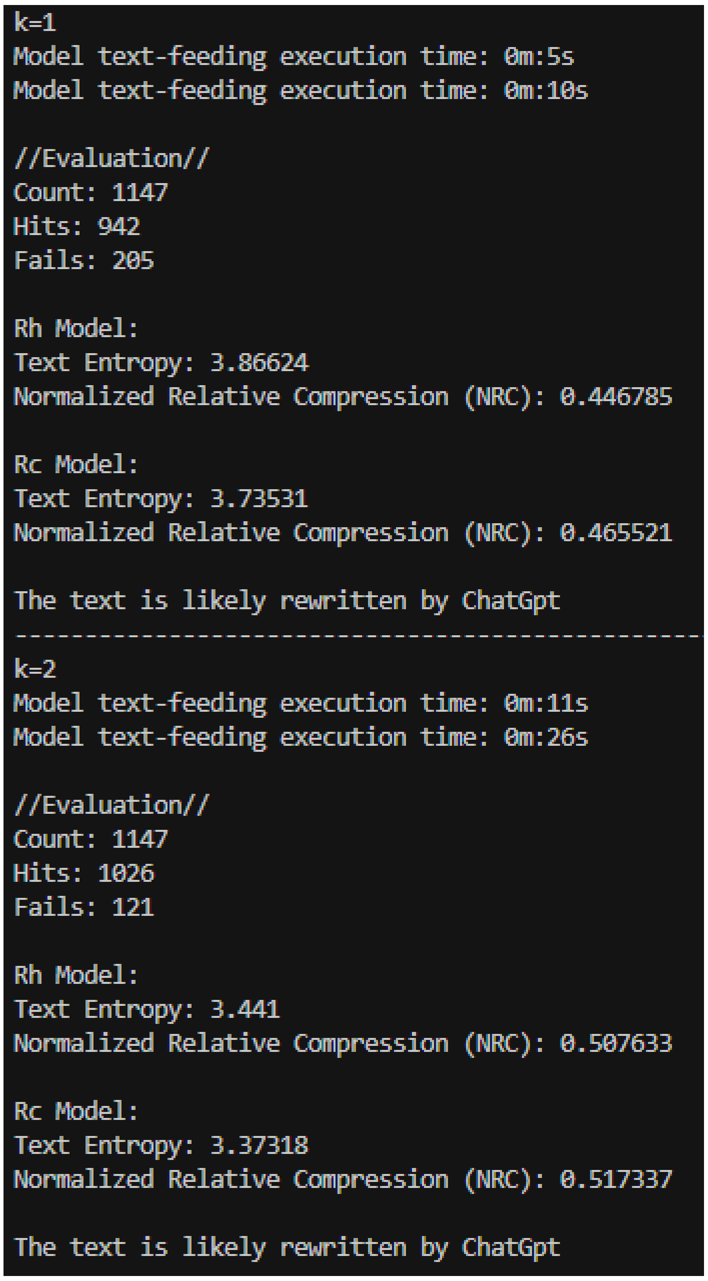
In the future, we’d like to be able to further optimise the program and be able to increase k past 14 to get better tests and a better understanding of how this can affect performance. On top of that, it would be interesting to disregard static values of k entirely and base the context word by word instead, or take partial sentences by using multiple words as context, similar to a language model, and predicting words instead of characters.

References

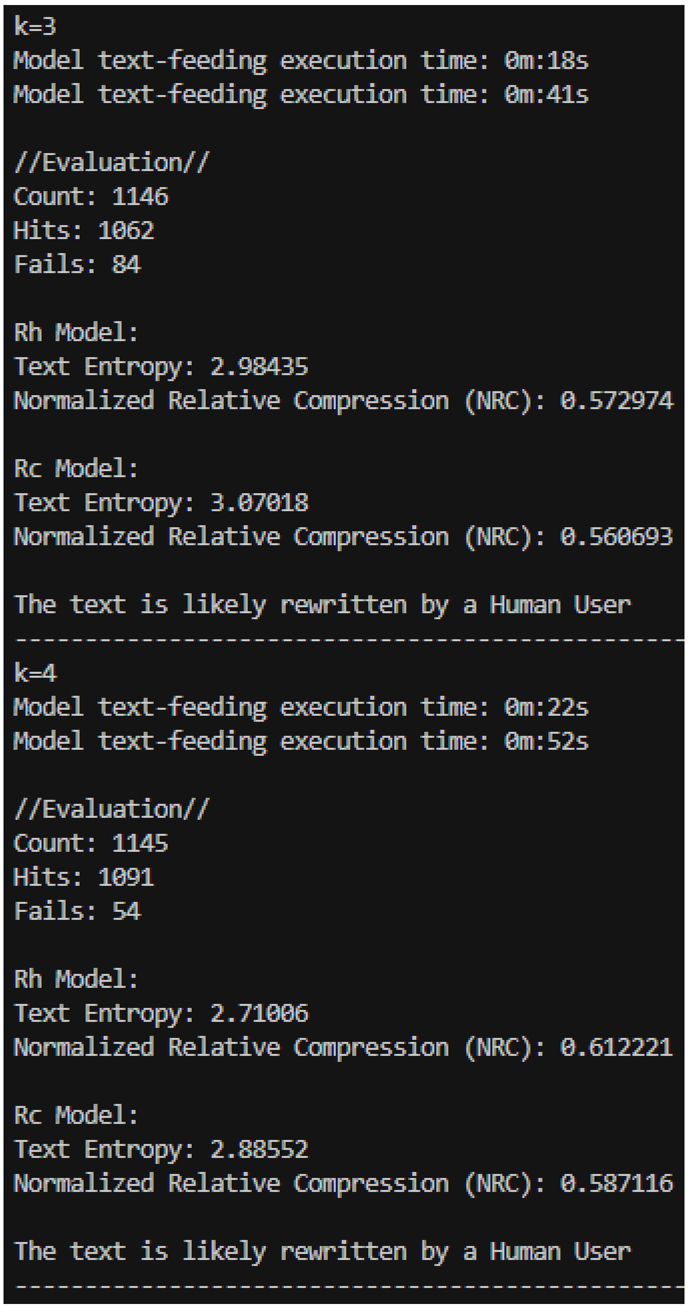
https://www.kaggle.com/datasets/jdragonxherrera/augmented-data-for-llm-detect-ai-generated-text/versions/1

Appendix

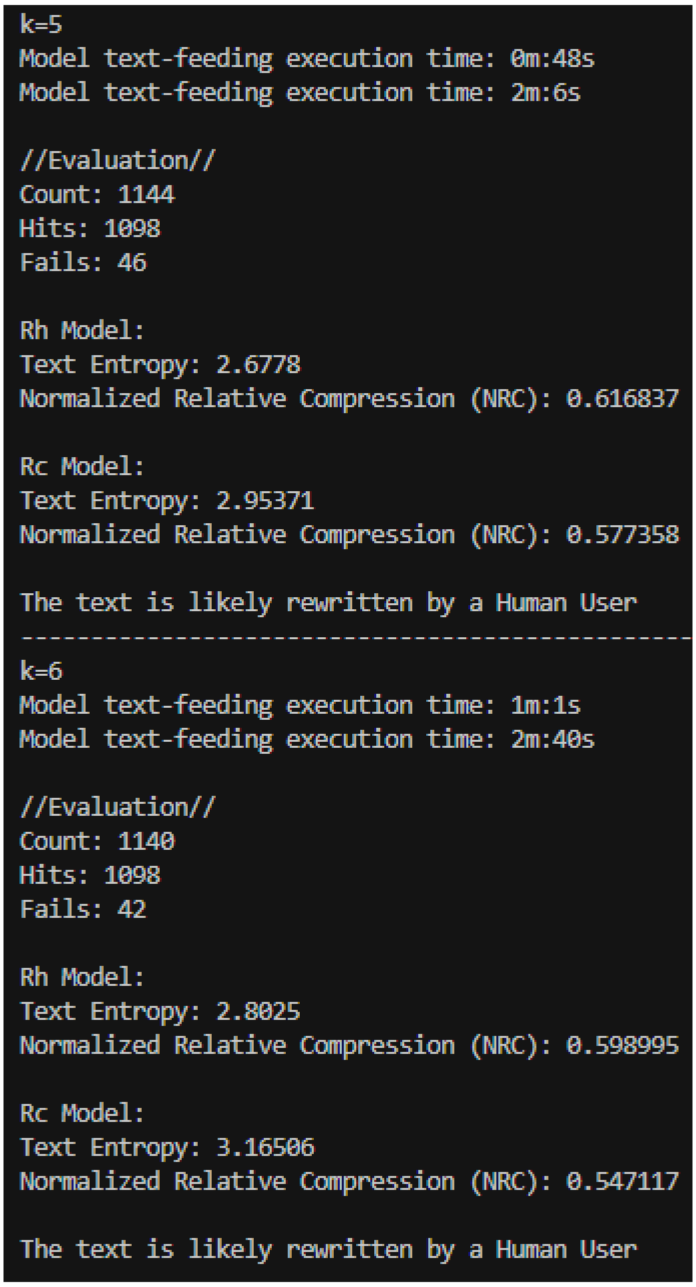
Below are the results of tests done by changing both k and α.



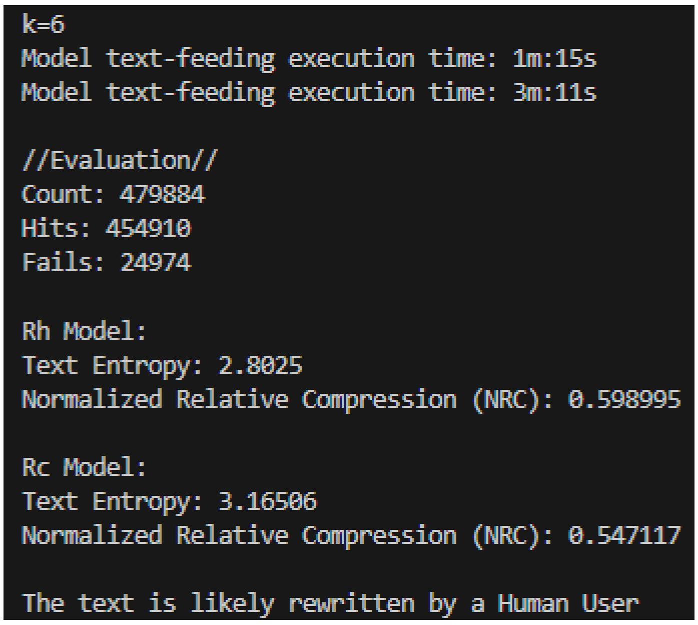
```
k=1
Model text-feeding execution time: 0m:5s
Model text-feeding execution time: 0m:10s

//Evaluation//
Count: 1147
Hits: 942
Fails: 205

Rh Model:
Text Entropy: 3.86624
Normalized Relative Compression (NRC): 0.446785

Rc Model:
Text Entropy: 3.73531
Normalized Relative Compression (NRC): 0.465521

The text is likely rewritten by ChatGpt
-------------------------------------------------
k=2
Model text-feeding execution time: 0m:11s
Model text-feeding execution time: 0m:26s

//Evaluation//
Count: 1147
Hits: 1026
Fails: 121

Rh Model:
Text Entropy: 3.441
Normalized Relative Compression (NRC): 0.507633

Rc Model:
Text Entropy: 3.37318
Normalized Relative Compression (NRC): 0.517337

The text is likely rewritten by ChatGpt
```



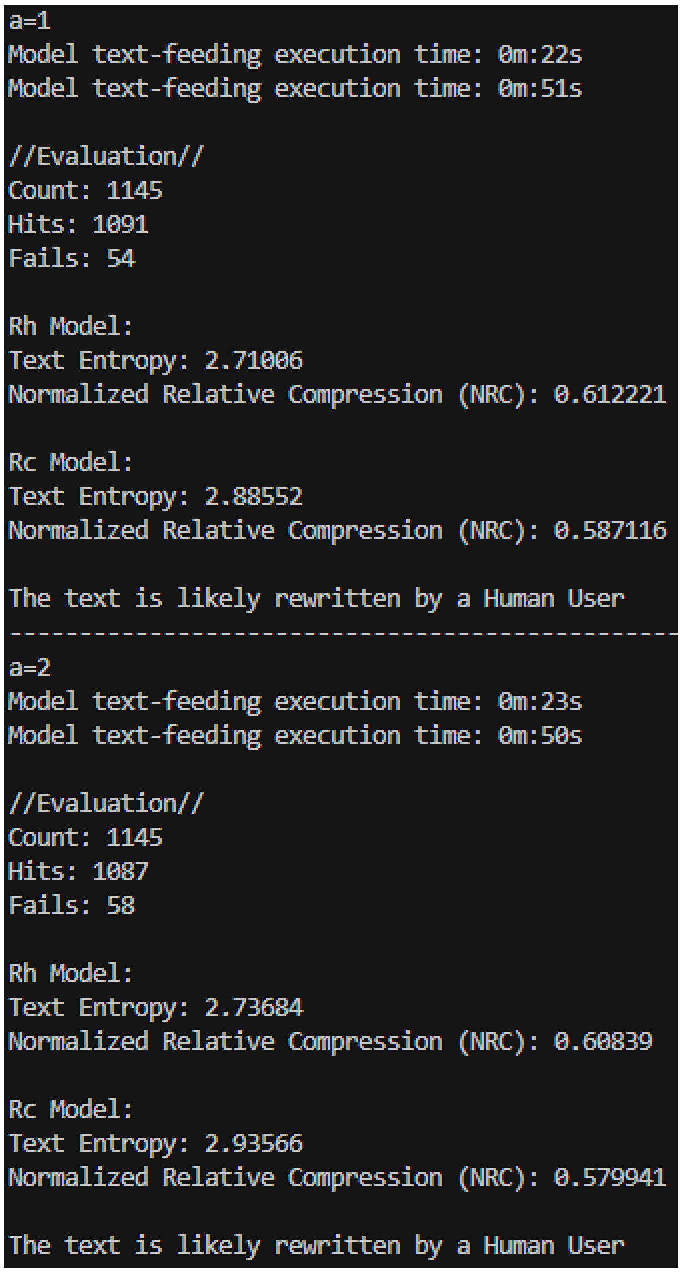
```
k=3
Model text-feeding execution time: 0m:18s
Model text-feeding execution time: 0m:41s

//Evaluation//
Count: 1146
Hits: 1062
Fails: 84

Rh Model:
Text Entropy: 2.98435
Normalized Relative Compression (NRC): 0.572974

Rc Model:
Text Entropy: 3.07018
Normalized Relative Compression (NRC): 0.560693

The text is likely rewritten by a Human User
------------------------------------------------
k=4
Model text-feeding execution time: 0m:22s
Model text-feeding execution time: 0m:52s

//Evaluation//
Count: 1145
Hits: 1091
Fails: 54

Rh Model:
Text Entropy: 2.71006
Normalized Relative Compression (NRC): 0.612221

Rc Model:
Text Entropy: 2.88552
Normalized Relative Compression (NRC): 0.587116

The text is likely rewritten by a Human User
------------------------------------------------
```



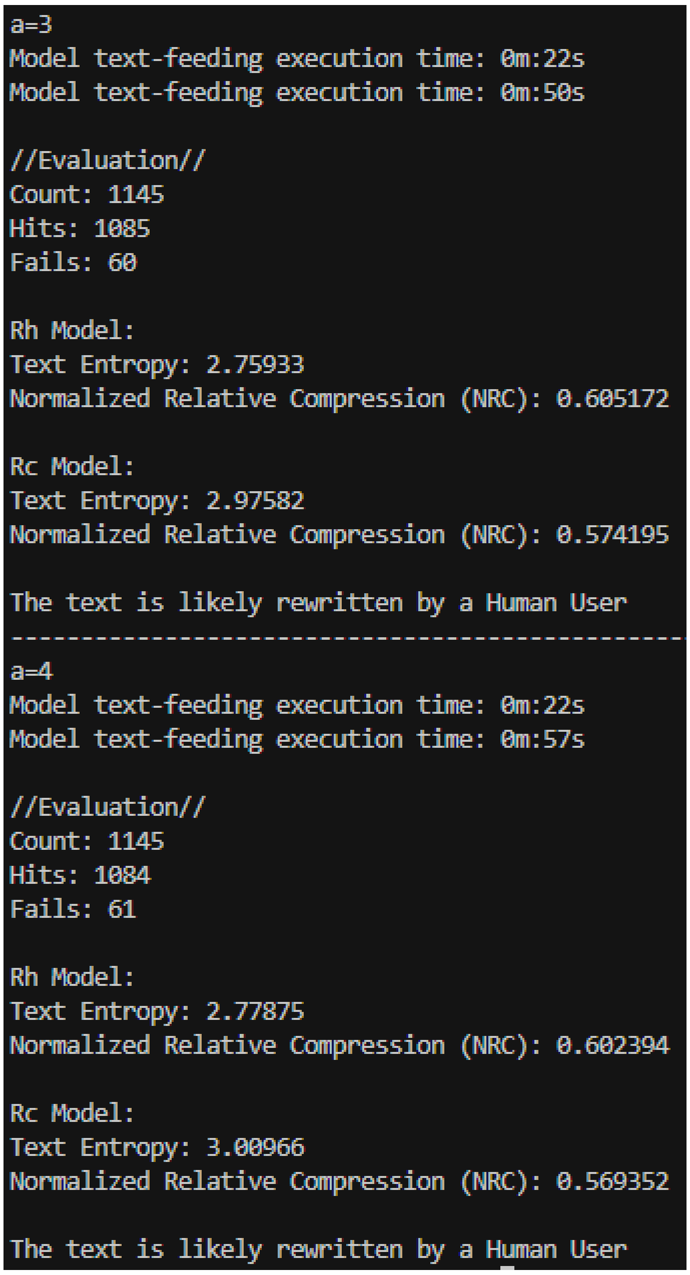
```
k=5
Model text-feeding execution time: 0m:48s
Model text-feeding execution time: 2m:6s

//Evaluation//
Count: 1144
Hits: 1098
Fails: 46

Rh Model:
Text Entropy: 2.6778
Normalized Relative Compression (NRC): 0.616837

Rc Model:
Text Entropy: 2.95371
Normalized Relative Compression (NRC): 0.577358

The text is likely rewritten by a Human User
------------------------------------------------
k=6
Model text-feeding execution time: 1m:1s
Model text-feeding execution time: 2m:40s

//Evaluation//
Count: 1140
Hits: 1098
Fails: 42

Rh Model:
Text Entropy: 2.8025
Normalized Relative Compression (NRC): 0.598995

Rc Model:
Text Entropy: 3.16506
Normalized Relative Compression (NRC): 0.547117

The text is likely rewritten by a Human User
```



```
k=6
Model text-feeding execution time: 1m:15s
Model text-feeding execution time: 3m:11s

//Evaluation//
Count: 479884
Hits: 454910
Fails: 24974

Rh Model:
Text Entropy: 2.8025
Normalized Relative Compression (NRC): 0.598995

Rc Model:
Text Entropy: 3.16506
Normalized Relative Compression (NRC): 0.547117

The text is likely rewritten by a Human User
```



```
a=1
Model text-feeding execution time: 0m:22s
Model text-feeding execution time: 0m:51s

//Evaluation//
Count: 1145
Hits: 1091
Fails: 54

Rh Model:
Text Entropy: 2.71006
Normalized Relative Compression (NRC): 0.612221

Rc Model:
Text Entropy: 2.88552
Normalized Relative Compression (NRC): 0.587116

The text is likely rewritten by a Human User
------------------------------------------------
a=2
Model text-feeding execution time: 0m:23s
Model text-feeding execution time: 0m:50s

//Evaluation//
Count: 1145
Hits: 1087
Fails: 58

Rh Model:
Text Entropy: 2.73684
Normalized Relative Compression (NRC): 0.60839

Rc Model:
Text Entropy: 2.93566
Normalized Relative Compression (NRC): 0.579941

The text is likely rewritten by a Human User
```



```
a=3
Model text-feeding execution time: 0m:22s
Model text-feeding execution time: 0m:50s

//Evaluation//
Count: 1145
Hits: 1085
Fails: 60

Rh Model:
Text Entropy: 2.75933
Normalized Relative Compression (NRC): 0.605172

Rc Model:
Text Entropy: 2.97582
Normalized Relative Compression (NRC): 0.574195

The text is likely rewritten by a Human User
-------------------------------------------------
a=4
Model text-feeding execution time: 0m:22s
Model text-feeding execution time: 0m:57s

//Evaluation//
Count: 1145
Hits: 1084
Fails: 61

Rh Model:
Text Entropy: 2.77875
Normalized Relative Compression (NRC): 0.602394

Rc Model:
Text Entropy: 3.00966
Normalized Relative Compression (NRC): 0.569352

The text is likely rewritten by a Human User
```